

CS451

Texturing

1

Jyh-Ming Lien

Department of Computer Science

George Mason University

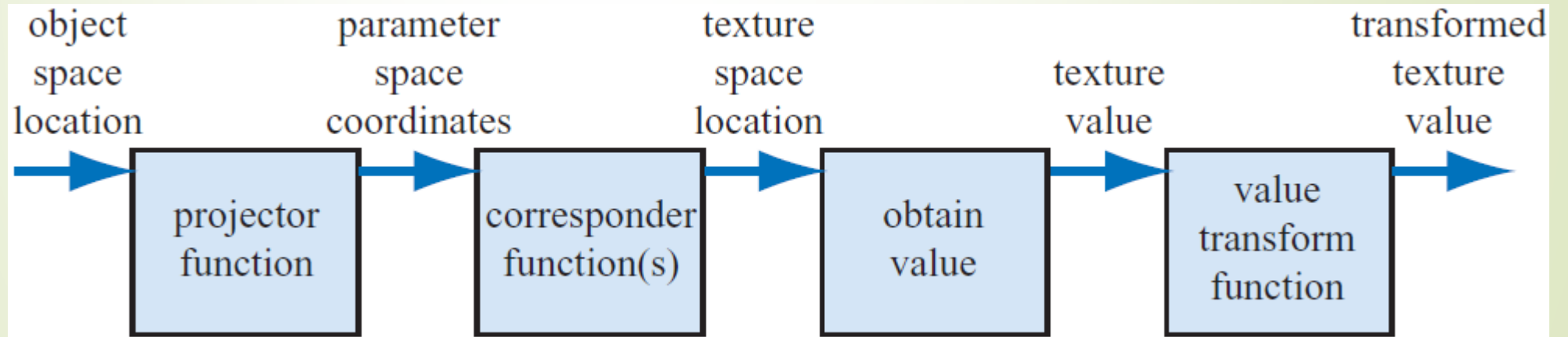
Texturing: Glue n-dimensional images onto geometrical objects

- More realism, and this is a cheap way to do it
 - Bump mapping
 - Plus, we can do environment mapping
 - And other things

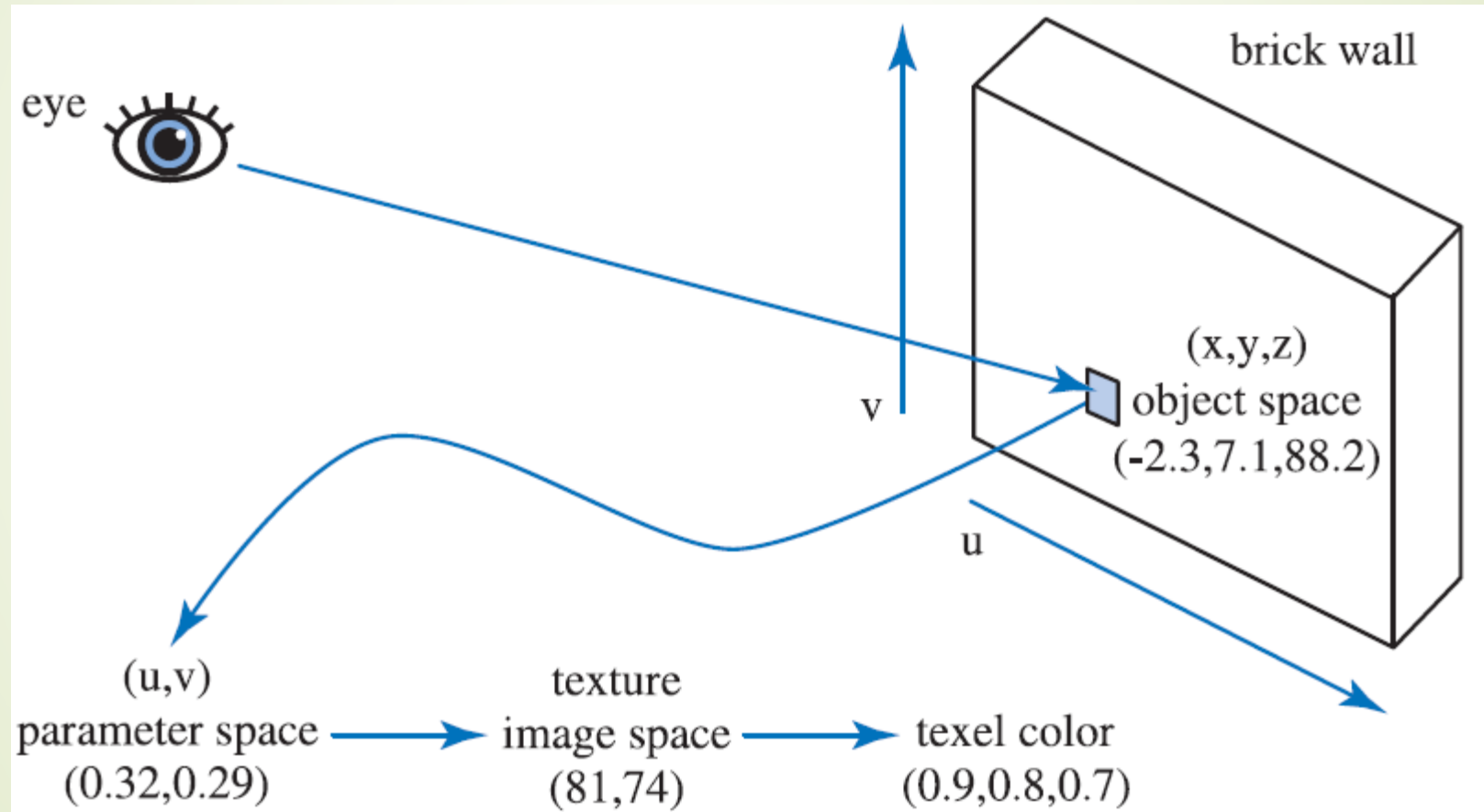


- We use triangles in real-time rendering, why?
 - Interpolation is rotation-invariant!
 - Not so for quadrilaterals.

Texture Pipeline

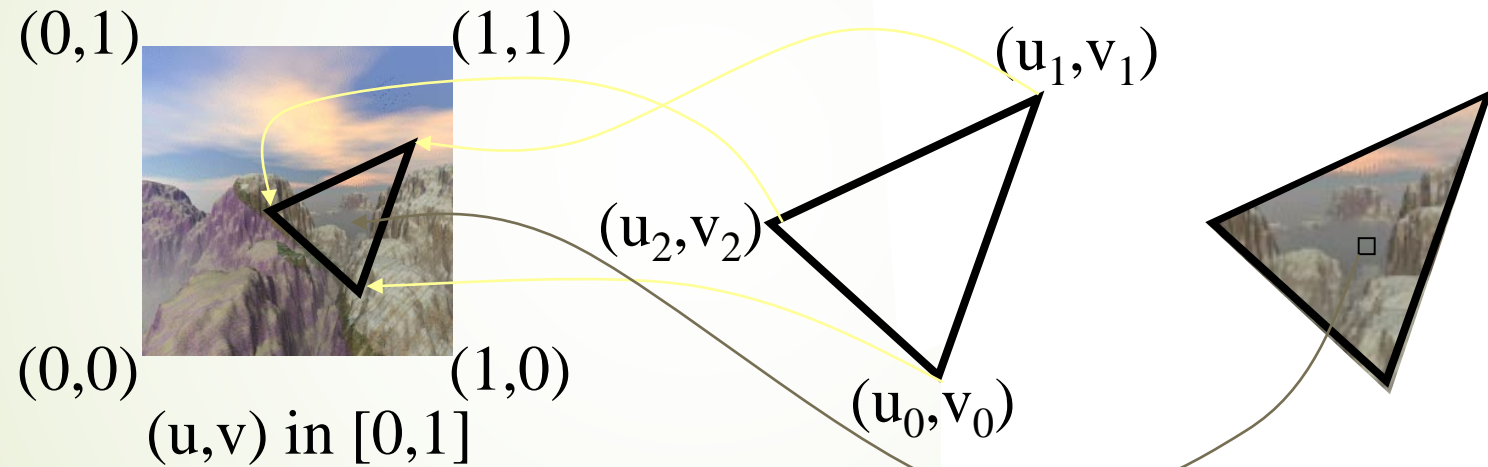


Texture Pipeline



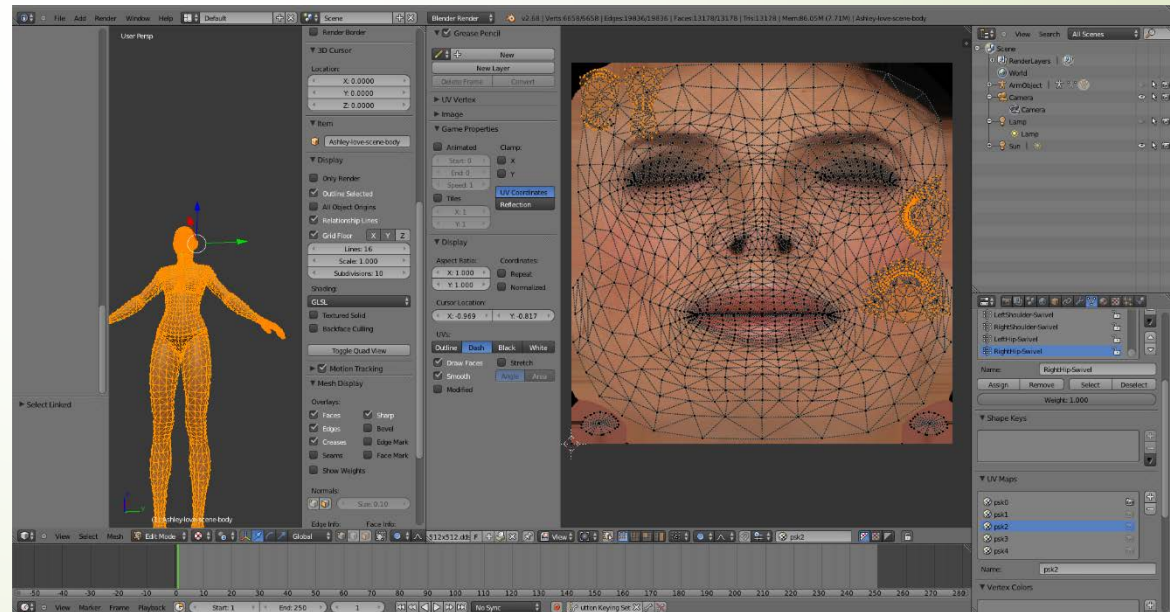
Texture coordinates

- How do you come up with these coordinates?



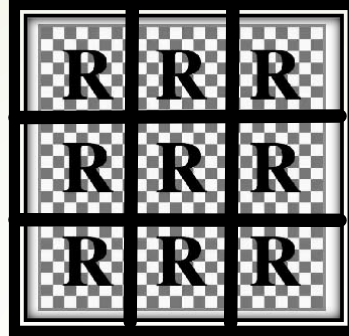
Projector Function

- ▶ Project surface point to parameter space, a.k.a. (u,v) space
- ▶ Projection is usually done automatically via
 - ▶ projector functions
 - ▶ Mesh unwarping algorithms
- ▶ Artists can edit (u,v) coordinates (a.k.a. **uv map editing**)



Projector Functions

Given an image:

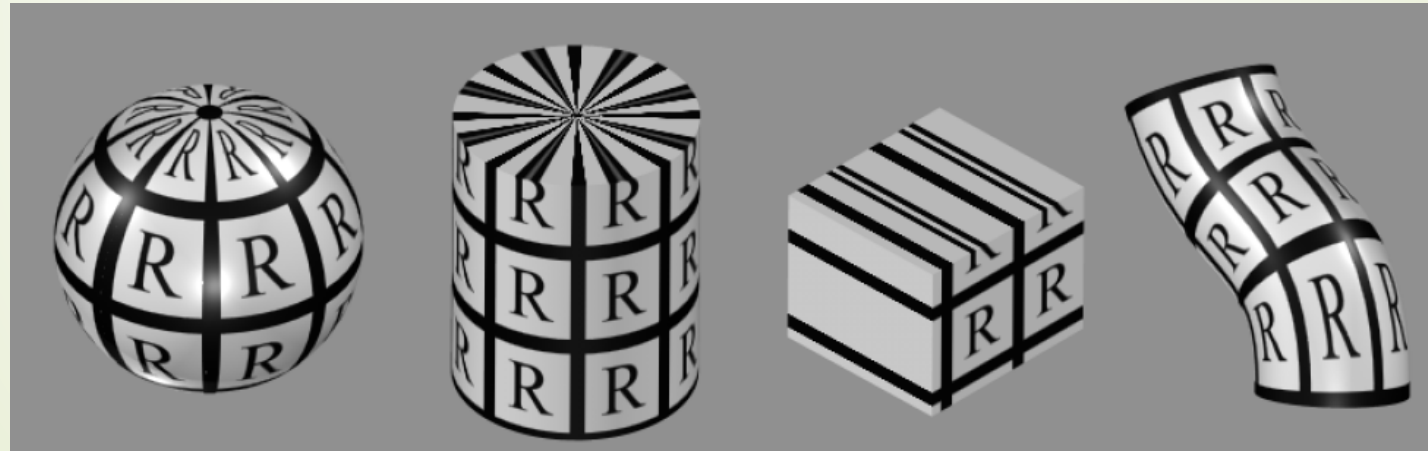


Spherical
projection

Cylindrical
projection

Planar
projection

Natural
projection



Projector Functions

- Different projections on the same shape

Spherical
projection

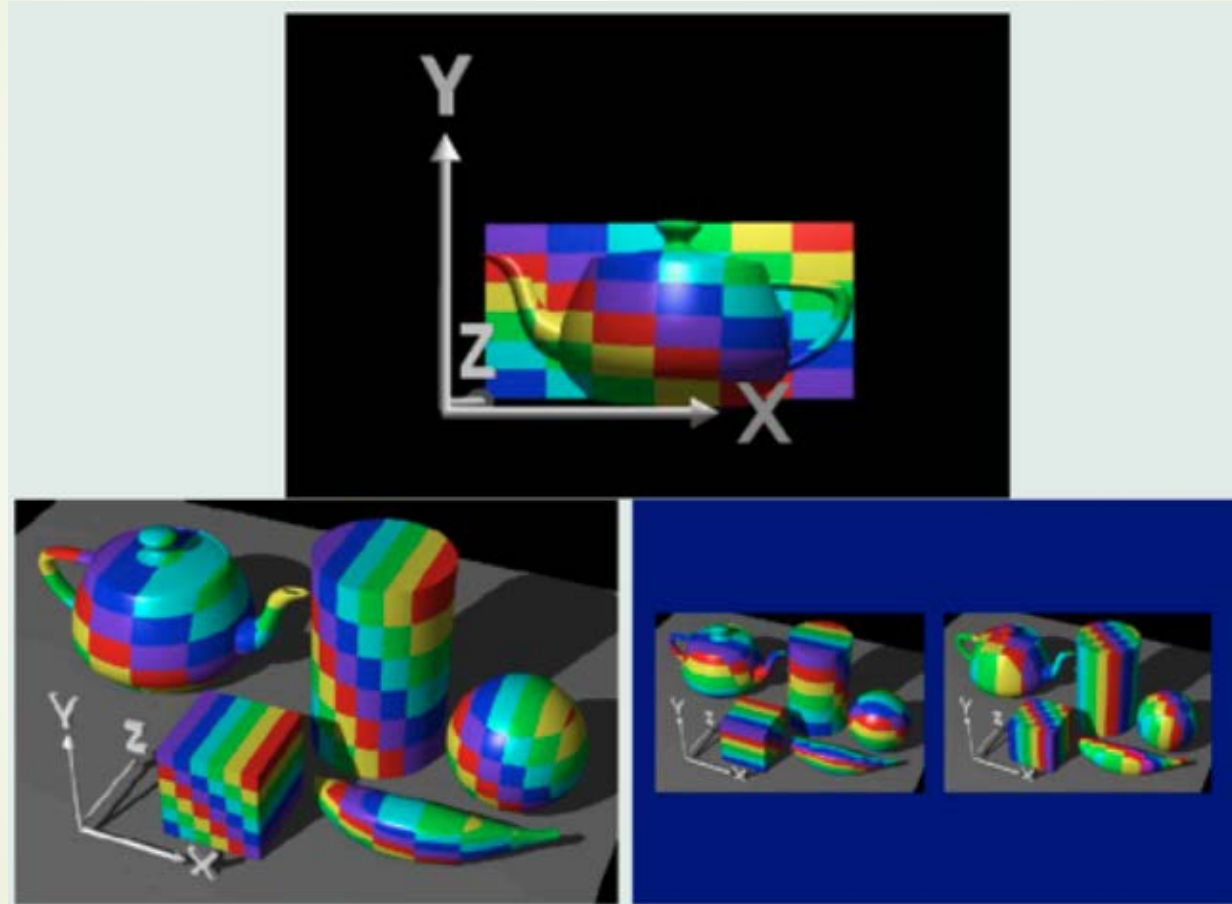
Cylindrical
projection

Planar
projection

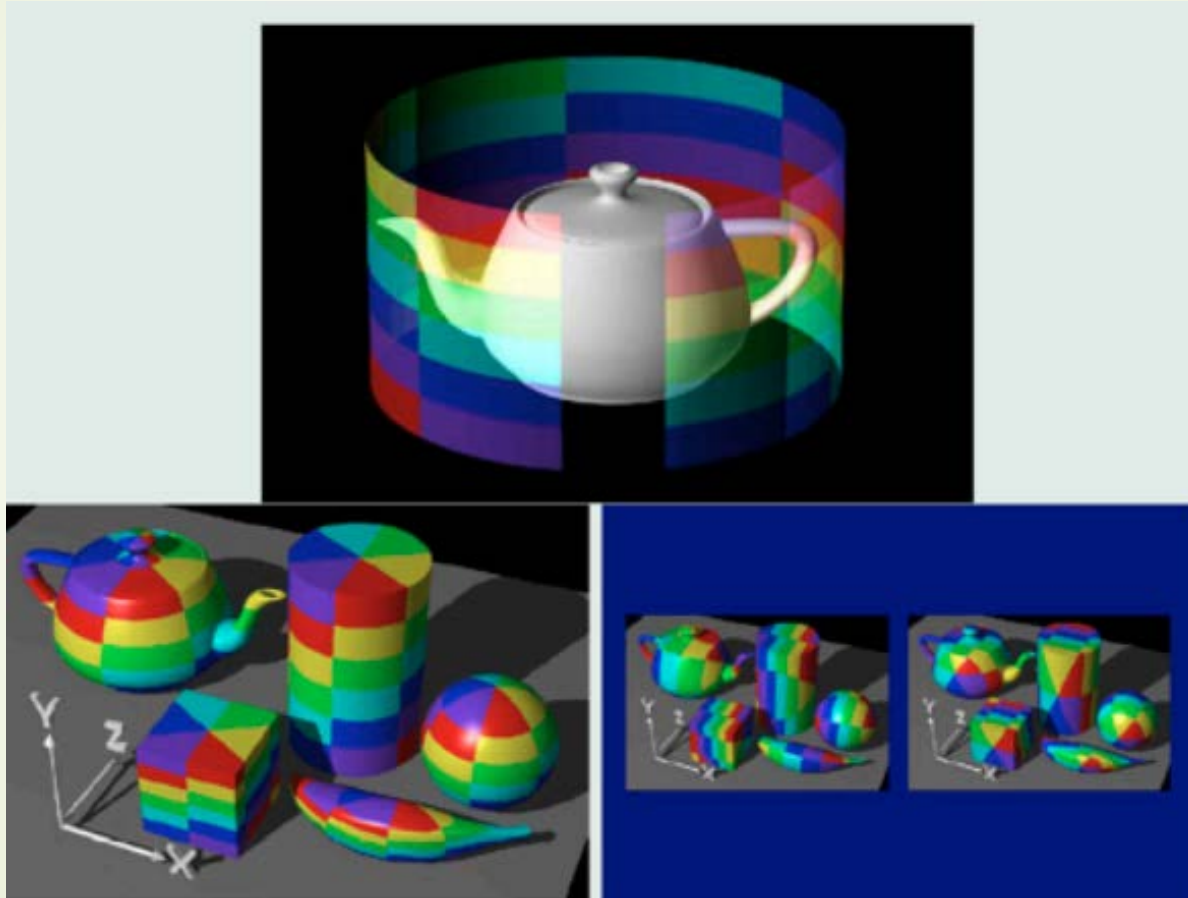
Natural
projection



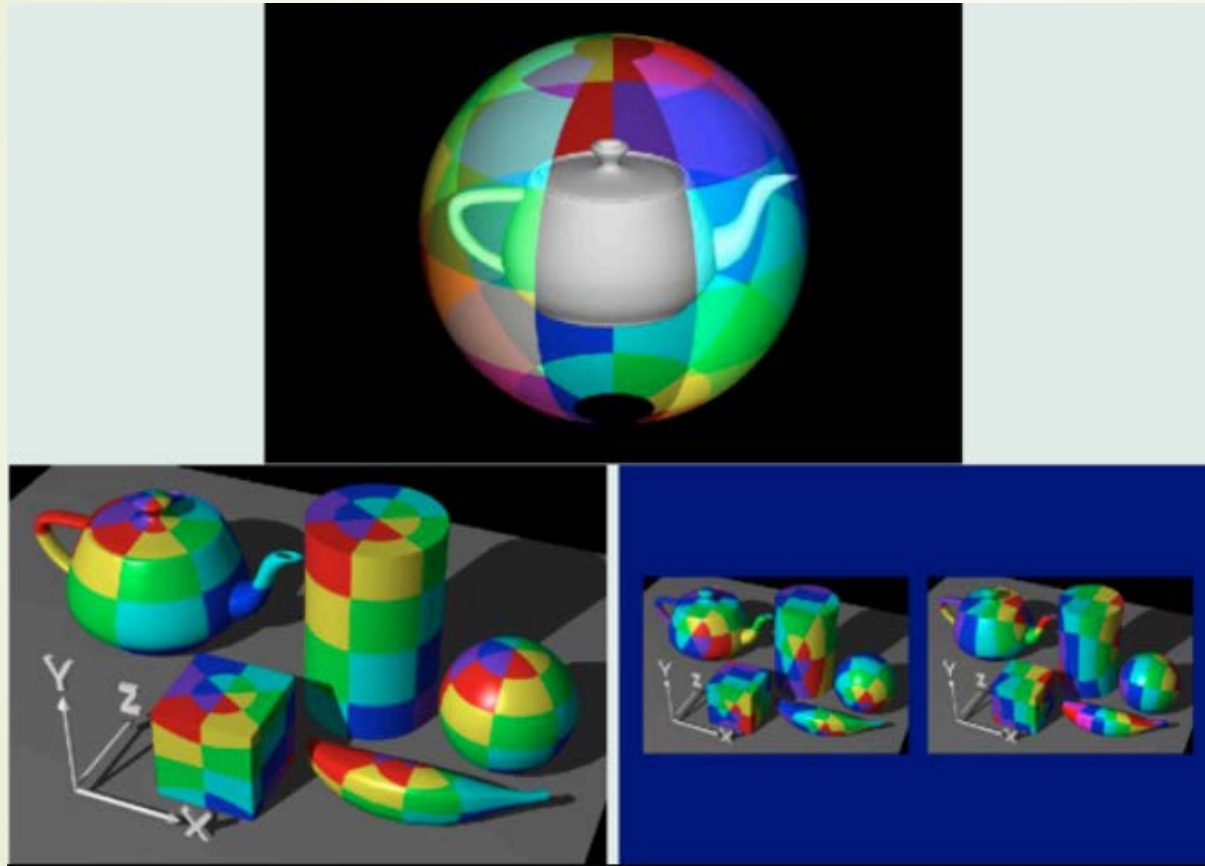
Planar



Cylindrical

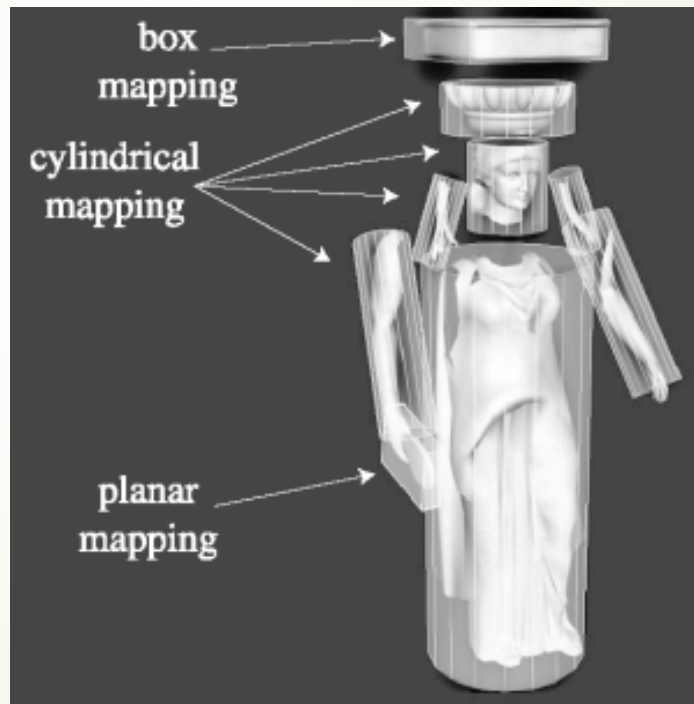


Spherical



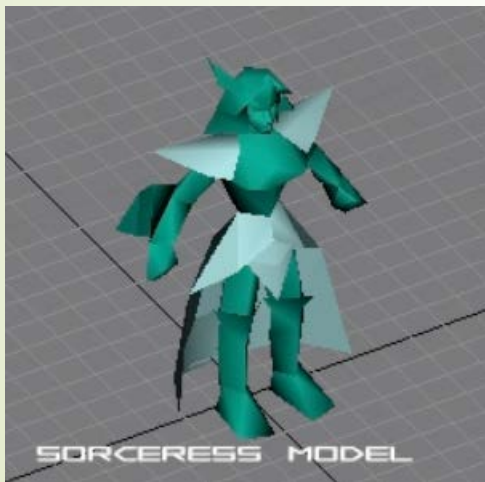
Projector Functions

- Various projector functions can be applied to the different parts of a model

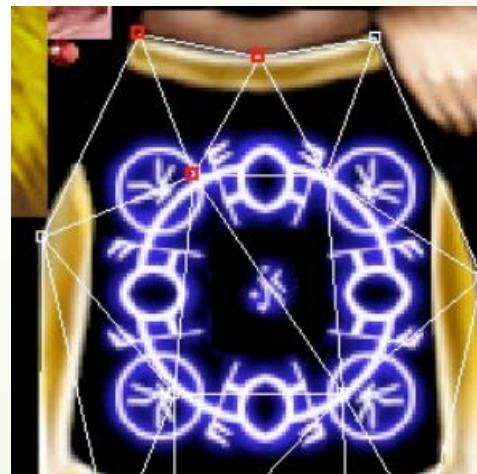


User Defined UV map

- ▶ Unwrap mesh
 - ▶ Set of planar projections
 - ▶ **Minimize distortion**
- ▶ Smaller textures for each of the projections
- ▶ Pack it into a larger texture

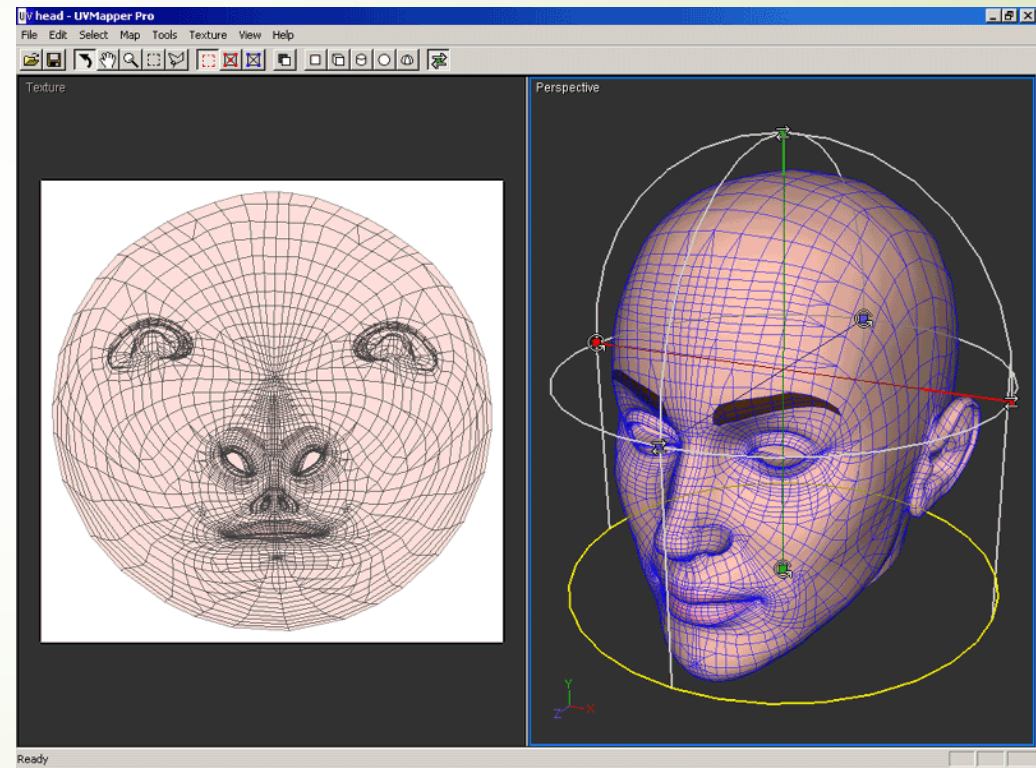


Warcraft III

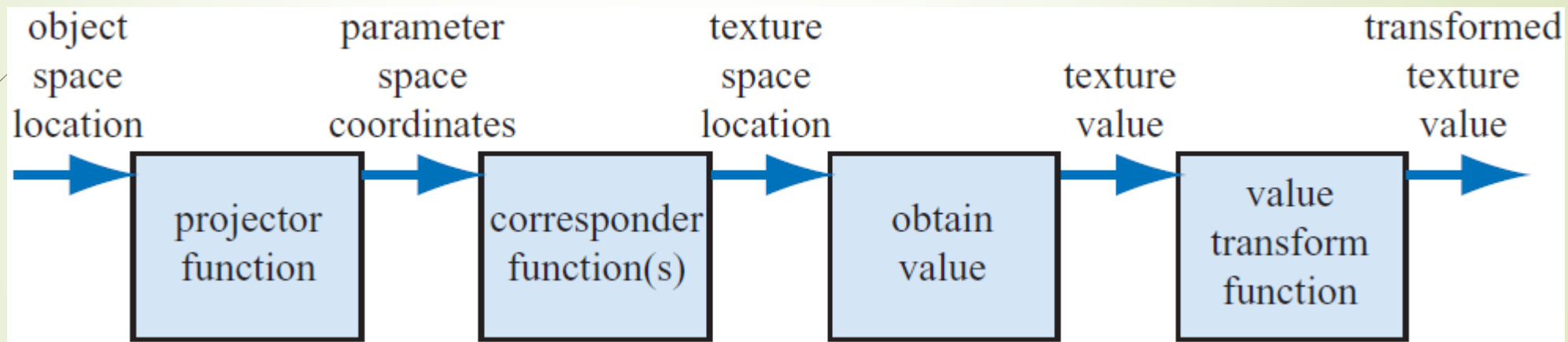


Demo: UVMapper

➔ <http://www.uvmapper.com/>

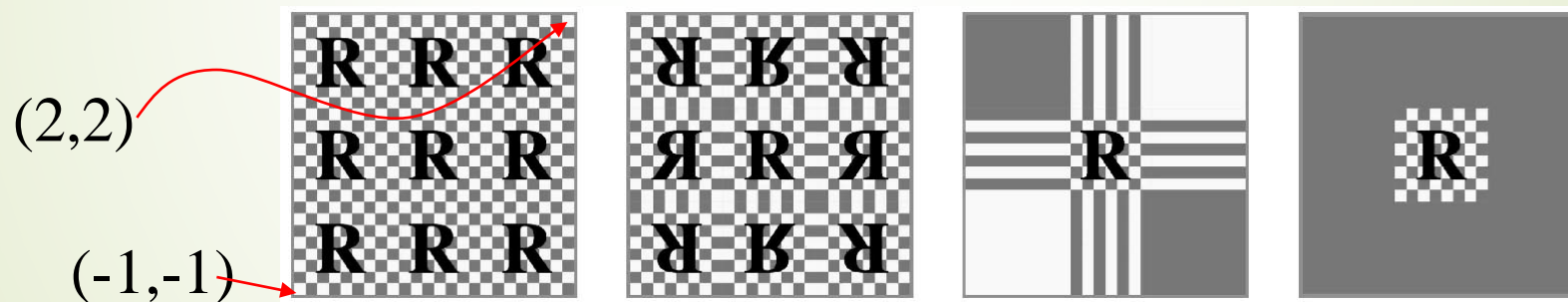


Texture Pipeline

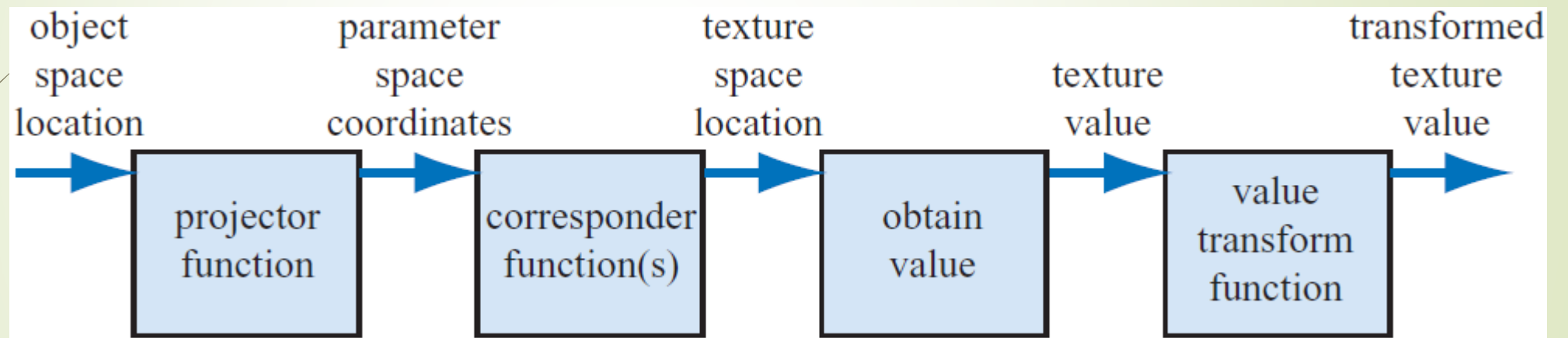


Corresponder Function

- ▶ What if $(u,v) > 1.0$ or < 0.0 ?
- ▶ To repeat textures, use just the fractional part
 - ▶ Example: $5.3 \rightarrow 0.3$
- ▶ Repeat, mirror, clamp, border:



Texture Pipeline



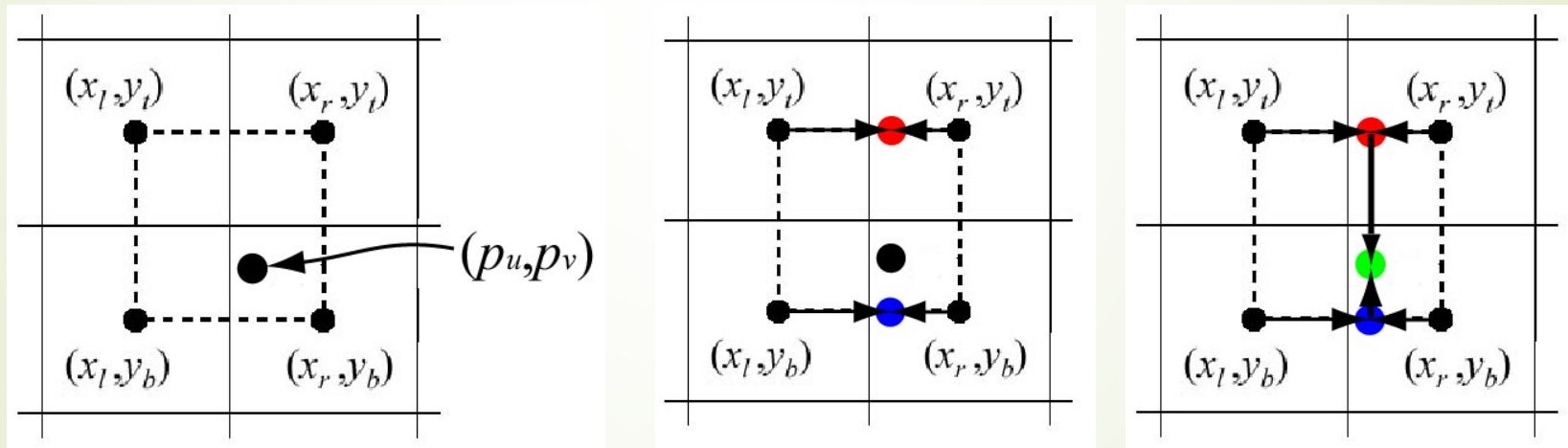
Obtain Value: Texture magnification

- Texture magnification of a 48x48 image on 320x320 pixel
- Box filter (nearest-neighbor) is poor in quality



Bilinear interpolation

- Texture coordinates (p_u, p_v) in $[0, 1]$
- Texture images size: $n \times m$ texels
- Nearest neighbor would access: $(\text{floor}(n \cdot u), \text{floor}(m \cdot v))$
- Interpolate 1D in x & y



Bilinear interpolation

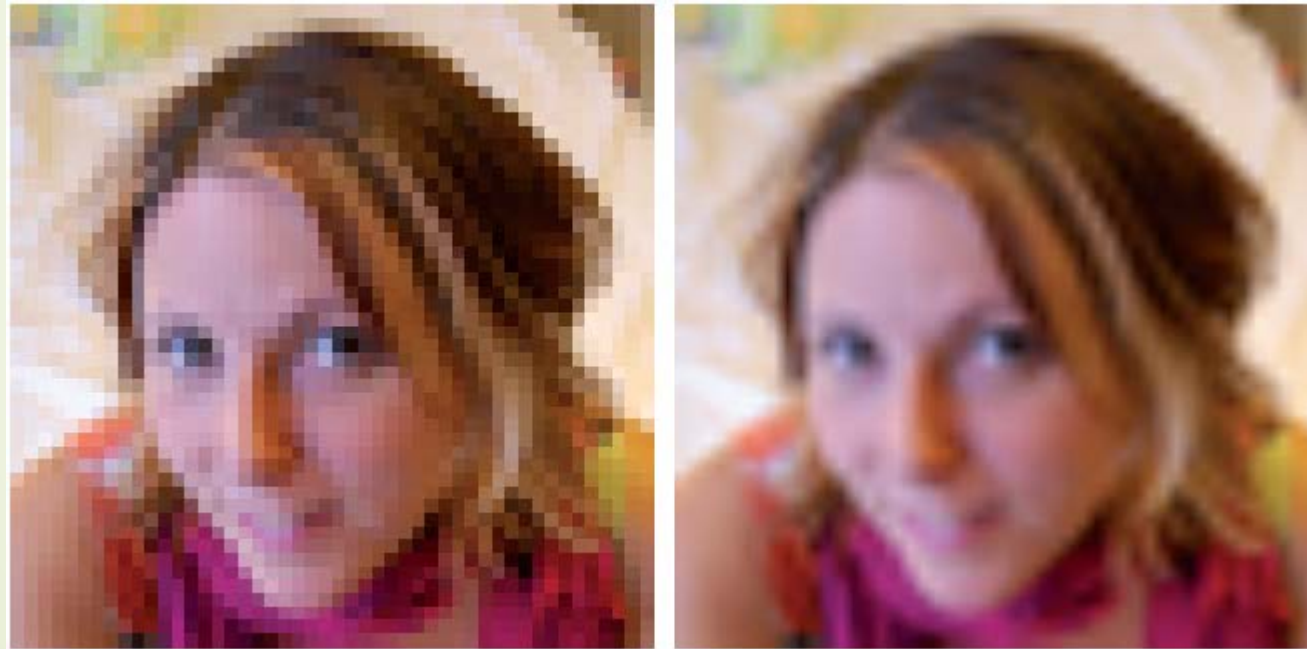
- ▶ Check out this formula at home
- ▶ $\mathbf{t}(u,v)$ accesses the texture map
- ▶ $\mathbf{b}(u,v)$ filtered texel

$$(u', v') = (p_u - \lfloor p_u \rfloor, p_v - \lfloor p_v \rfloor).$$

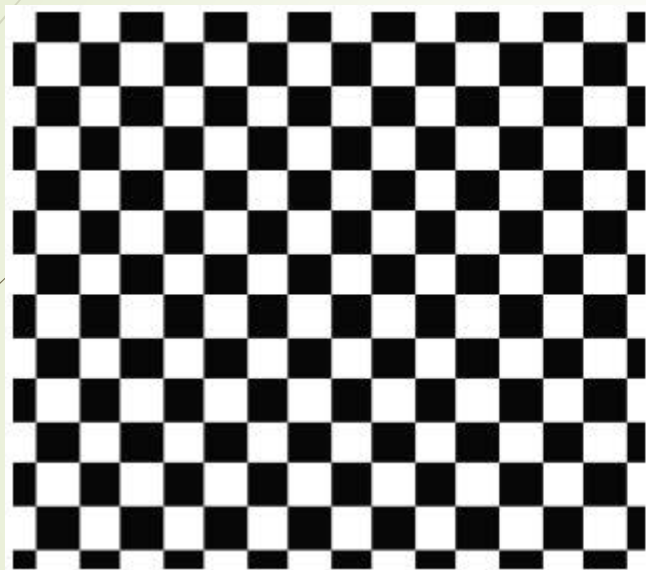
$$\begin{aligned} \mathbf{b}(p_u, p_v) = & (1 - u')(1 - v')\mathbf{t}(x_l, y_b) + u'(1 - v')\mathbf{t}(x_r, y_b) \\ & + (1 - u')v'\mathbf{t}(x_l, y_t) + u'v'\mathbf{t}(x_r, y_t). \end{aligned}$$

Bilinear interpolation

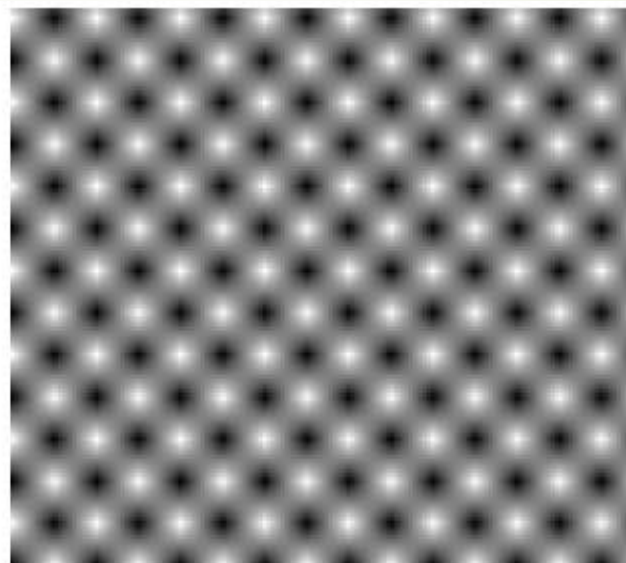
- Nearest neighbor filtering vs. Bilinear interpolation



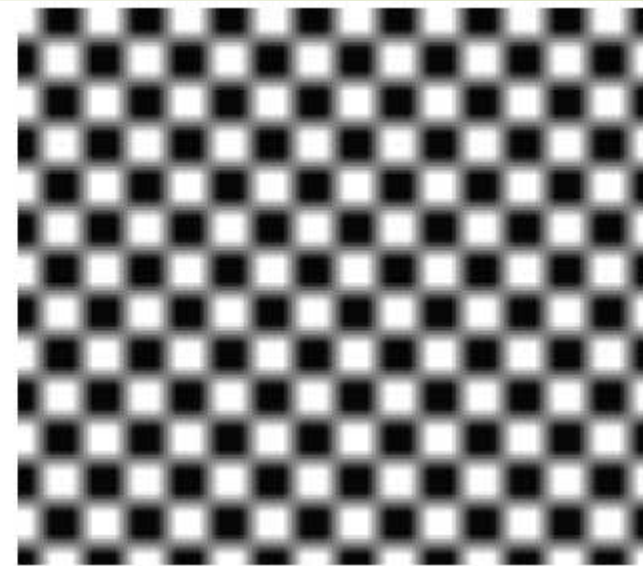
Problem with Bilinear interpolation



Nearest neighbor



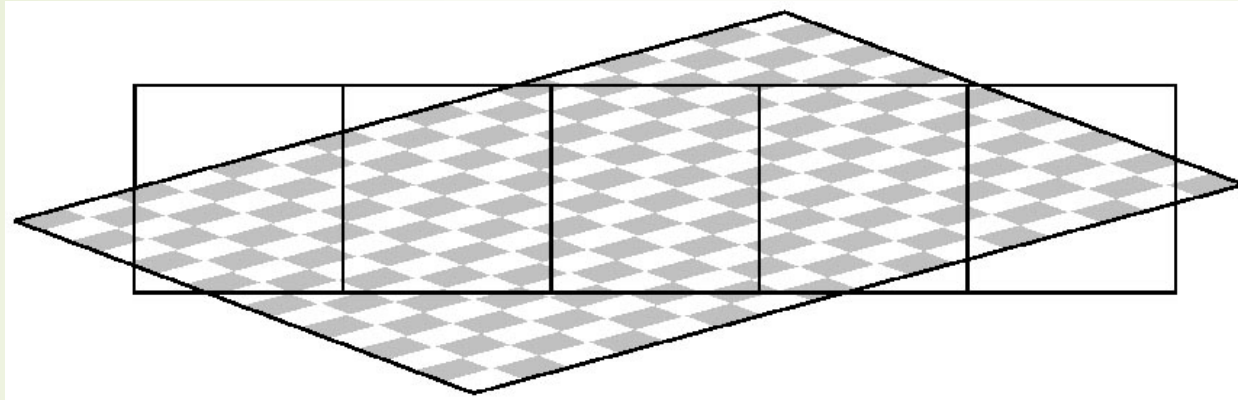
Bilinear interpolation



After remapping:
 $C > 0.6$ is white
 $C < 0.4$ is black

Texture minification

What does a pixel "see"?



- ▶ Several texels can be covered by a single pixel
 - ▶ Nearest neighbor (using the center of the pixel)
 - ▶ Bilinear interpolation (again, using the center of the pixel)
 - ▶ Compute an average of all enclosed texels
 - ▶ Works better but can be slow

Texture minification

